UNIVERSITÀ DI PARMA
Dipartimento di Ingegneria e Architettura

Identification (Entity Authentication)

Luca Veltri

(mail.to: luca.veltri@unipr.it)

Course of Cybersecurity, 2022/2023

http://netsec.unipr.it/veltri

Entity Authentication - Identification

- Techniques designed to allow one party (the verifier or authenticator) to gain assurances that the identity of another (the prover, claimant, or supplicant) is as declared
  - **preventing impersonation**
- A major difference between entity authentication and message authentication is that
  - **message authentication itself provides no timeliness guarantees with respect to when a message was created**
  - **whereas entity authentication involves proof of a claimant's identity through actual communications with an associated verifier during execution of the protocol itself**
    - provide assurances only at the particular instant in time of successful protocol completion
      - If ongoing assurances are required, additional measures may be necessary

Basis of identification

- Entity authentication techniques may be divided into three main categories, depending on which of the following the security is based:
  - **1. something known**
    - e.g. standard passwords (sometimes used to derive a symmetric key), Personal Identification Numbers (PINs), secret or private keys
  - **2. something possessed**
    - this is typically a physical accessory, as a “passport”
    - e.g. cards like magnetic-striped cards, chip cards (also called smart cards or IC cards), and hand-held customized calculators (passwd generators) which provide time-variant passwords
  - **3. something inherent to a human individual**
    - this category includes methods which make use of human physical characteristics and involuntary actions (biometrics), such as handwritten signatures, fingerprints, voice, retinal patterns, and dynamic keyboarding characteristics
    - these techniques are not further discussed

Characteristics of identification protocols

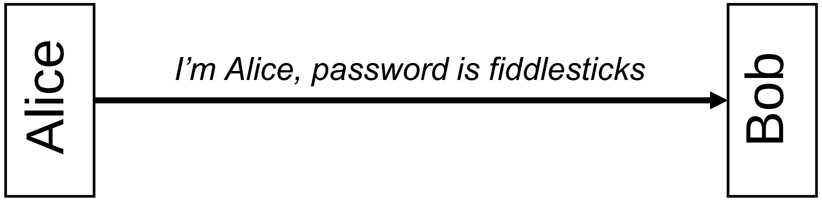
- Direction of the identification (reciprocity):
  - **unilateral identification**
    - only one party proves its identity to the other party
  - **mutual identification**
    - both parties may corroborate their identities to the other
- Computational and/or communication efficiency:
  - **computational efficiency**
    - the number of operations required to execute a protocol
  - **communication efficiency**
    - this includes the number of passes (message exchanges) and the bandwidth required (total number of bits transmitted)

Characteristics of identification protocols (cont.)

- Real-time involvement of a third party (if any):
  - an on-line trusted third party to distribute symmetric keys to communicating entities for authentication purposes; or
  - an on-line trusted third party to verify the authentication information sent by the supplicant, or
  - an on-line (untrusted) directory service for distributing public-key certificates
- Storage of secrets
  - ➢ **which information must be stored to perform verification**
    - symmetric secret
      - in clear or encrypted/hashed value
    - in case of public keys, they are not secret, however:
      - how obtaining public key of the peer-entity
      - how storing public key of the peer-entity
  - ➢ **where secrets are maintained/stored**
    - RAM
    - local disks
    - hardware tokens

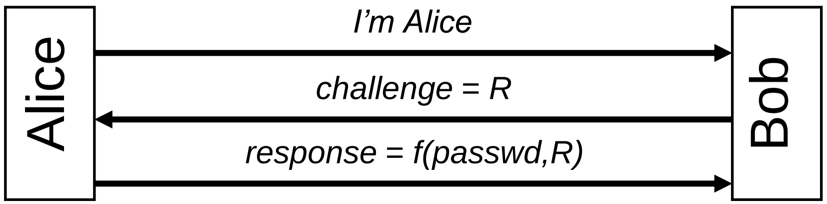
Basic authentication scheme

- Conventional (basic) authentication schemes consist in sending a fixed (time-invariant) symmetric secret
  - **shared secret between the user and system**
    - like a key or a password
  - **thus fall under the category of symmetric-key techniques**
- For example, to gain access to a system resource the user enters a (user id, password) pair
  - **weak authentication if used through an insecure channel**
  - **it is not associated to a given time (time-invariant)**
  - **the verifier can store the hash of the secret**



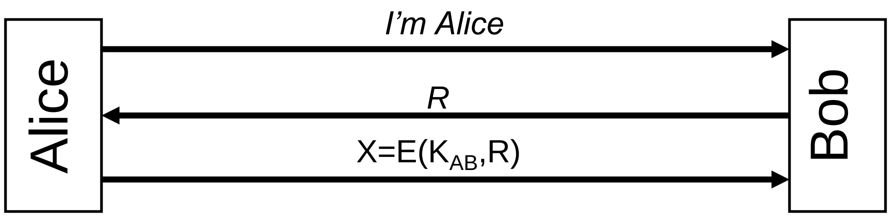
Challenge-response authentication

- The idea is that one entity (the claimant) “proves” its identity to another entity (the verifier) by demonstrating knowledge of a secret, without revealing the secret itself to the verifier
- This is done by providing a response to a *time-variant* challenge, where the response depends on both the challenge and the entity’s secret
  - ***time-variant*** **challenge is used to counteract replay and interleaving attacks**
    - a number, a text string, a bit string chosen randomly by one entity
    - a sequence number, incremented sequentially, a timestamp referring to a given time interval, optionally sent during the exchange
      - a validity interval of numbers may be considered
  - **the response is verified by using the same entity’s secret OR other information associated to the secret**
    - e.g. hash value, public key, etc.



Authentication with symmetric key

computes $E(K_{AB},R)$



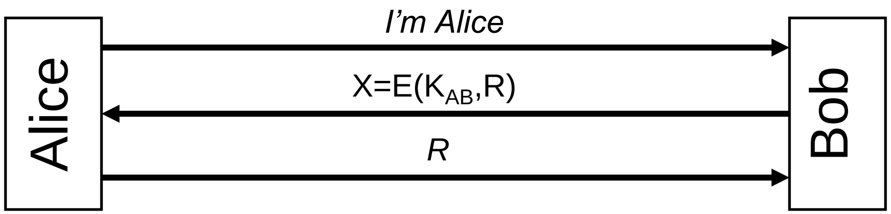
knows alice's secret

Picks random R

computes and compare the $E(K_{AB},R)$

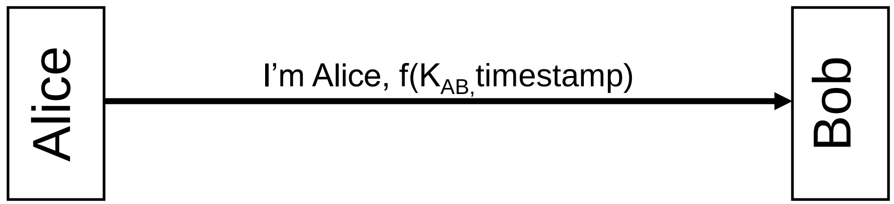
- Note:
  - **does not require reversible cryptography**
  - **function *E* can be replaced by one-way function of a secret and the challenge**
    - e.g. $X=MAC(K_{AB},R)$, or $X=H(secret_{AB} \| R)$
    - in general: $X=f(K_{AB},R)$
- drawbacks:
  - **an eavesdropper could mount an off-line password guessing attack**
  - **It requires that the authenticator maintain a copy of the password/key**
    - some who read the Bob's passwd-database (the authenticator) can later impersonate Alice (the claimant)

Authentication with symmetric key (variant 1)

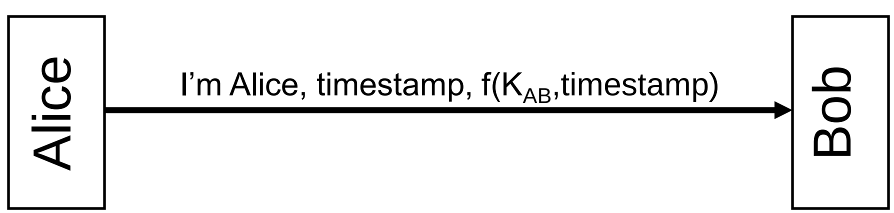


- differences:
  - **requires reversible cryptography**
    - e.g. $R=D(K_{AB},X)$
  - **if R is a recognizable quantity with limited lifetime (e.g. a random number concatenated with a timestamp), Alice can authenticate Bob**
  - **if R is a recognizable quantity, Carol can mount an offline passwd-guessing attack without eavesdropping**
    - Carol obtains $K_{AB}\{R\}$ (second message) by just sending the first message to Bob claiming to be Alice
      - – Carol doesn't need that Alice authenticates with Bob

Authentication with symmetric key (variant 2)

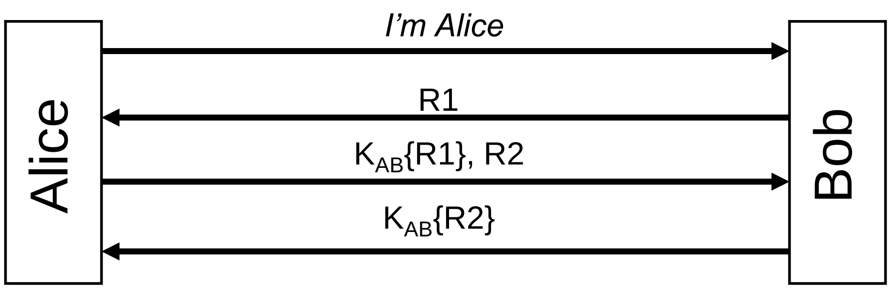


- differences:
  - **required only one message**
    - this mechanism can be added very easily to a protocol originally designed for sending passwd as cleartext
    - more efficient
  - **function f() does not require to be reversible**
  - **several pitfalls due to the time validity (time synchronization between Alice and Bob, authentication with multiple server with the same passwd, etc)**
- variant:



Mutual authentication with symmetric key

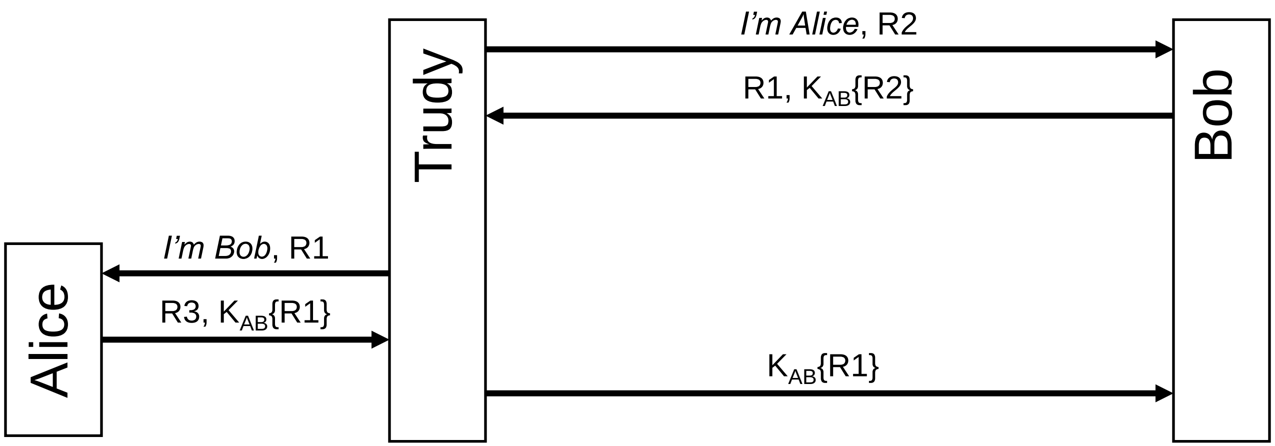
- or shorter:



Mutual authentication with symmetric key

- or shorter:

- however, possible reflection attack:



- Good general principles of authentication protocols:
  - **the initiator should be the first to prove its identity**
  - **messages sent in opposite directions should differ**

Authentication with public key

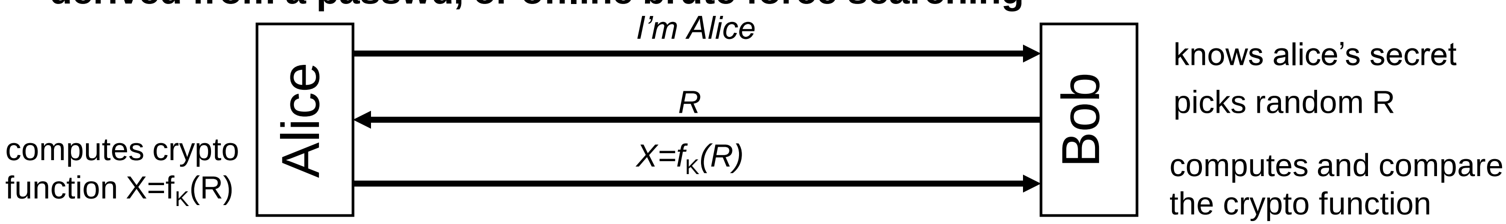
- or

- property:
  - **the database at Bob must not be protected from reading**
    - must be protected only for unauthorized modification (integrity protection)
- drawbacks:
  - **if you can trick Alice into signing or decrypting something, you can impersonate Alice (first and second scheme, respectively)**
  - **by asking Alice to authenticate, you can obtain a signature or decryption**
- countermeasures:
  - **not use the same key for two different purposes unless the design for all uses are coordinated (this is a general rule), and/or**
  - **impose enough structure to be signed (nonce, realm, timestamp, etc.)**

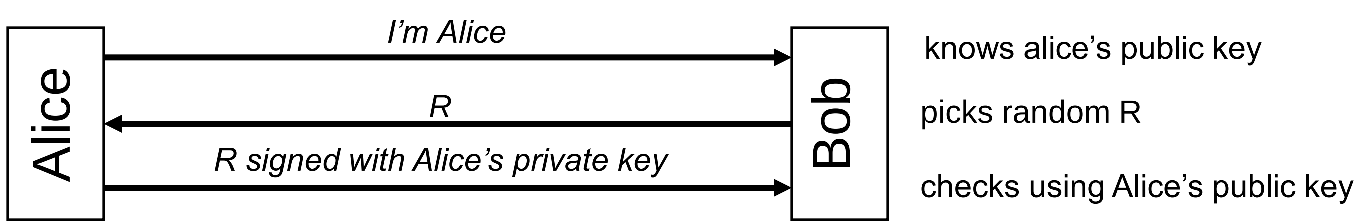
Eavesdropping and server database reading

- Protection against server database reading:
  - **vulnerable to eavesdropping**

- Protection against eavesdropping:
  - **vulnerable to database reading, and to offline password guessing if the secret (key) is derived from a passwd, or offline brute force searching**



- Protection against both using asymmetric cryptography:



One-time passwords

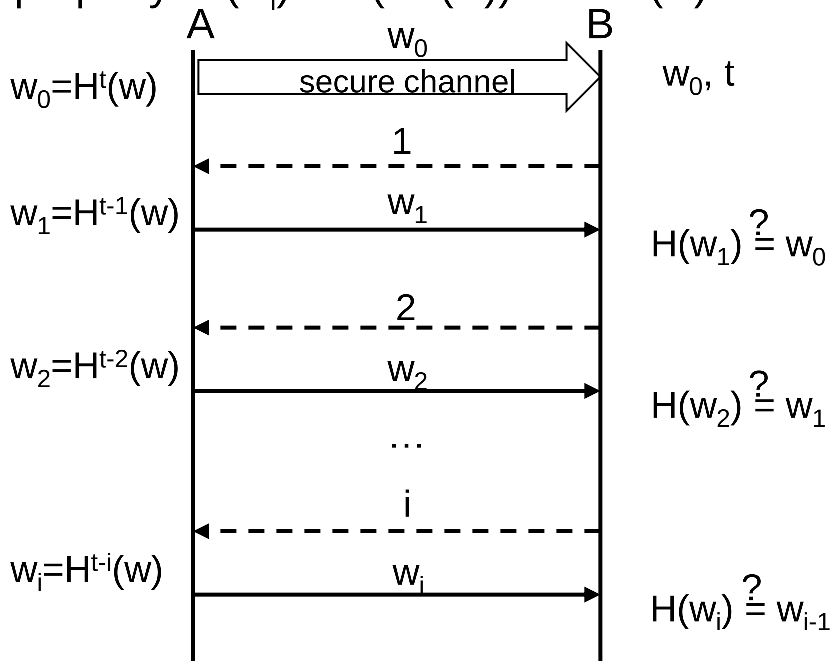
- An alternative to use fixed secrets/passwords is using one-time secrets/passwords
  - **each password is used only once**
  - **such schemes are safe from passive adversaries who eavesdrop and later attempt impersonation**
- Can be easily implemented in Smart/token Cards

One-time passwords (cont.)

- Some one-time passwords variations:
  - **shared lists of one-time passwords**
    - use a sequence or set of secret passwords, (each valid for a single authentication), distributed as a pre-shared list
    - if the list is not used sequentially, the system may check the entered password against all remaining unused passwords
    - a variation involves use of a challenge-response table
    - a drawback is maintenance of the shared list
  - **sequentially updated one-time passwords**
    - during authentication using password *i*, the user creates and transmits to the system a new password (password *i+1*) encrypted under a key derived from password *i*
    - this method becomes difficult if communication failures occur
  - **one-time password sequences based on a one-way function**
    - more efficient than the previous one
    - may be viewed as a challenge-response protocol where challenge is implicitly defined by the current position within the pwd sequence

Lamport's scheme

- Simple One-Time Password (OTP) mechanism
  - **does not require the server to maintain a shared secret nor a list of hashes**
  - **based on a secret *w* and a one-way function *H*, the sequence of *t* passwords *H(w), H(H(w)), … , $H^t(w)$* is defined**
  - **these passwords are used in the reverse order**
    - the authenticator is initialized with $w_0=H^t(w)$
    - password for the i^{th} identification exchange ($1 \le i \le t$) is: $w_i = H^{t-i}(w)$
    - resulting property: $H(w_i) = H(H^{t-i}(w)) = H^{t-i+1}(w) = H^{t-(i-1)}(w) \equiv w_{i-1}$



Lamport's scheme (cont.)

- Lamport's scheme usage:
  - **user A begins with a secret *w* and a constant *t* (e.g., *t* = 100 or 1000), defining the number of identifications to be allowed**
  - **A transfers (the initial shared secret) $w_0 = H^t(w)$, in a manner guaranteeing its authenticity, to the system B**
  - **B initializes its counter i_A for A to 1 ($i_A = 1$)**
  - **A $\rightarrow$ B : A, i, $w_i = H^{t-i}(w)$**
    - w_i is easily computed either from w or from an appropriate intermediate value saved during the computation of $H^t(w)$ initially
    - B checks that $i = i_A$, and that the received password w_i satisfies: $H(w_i) = w_{i-1}$

Zero-knowledge identification protocols

- Zero-knowledge (ZK) protocols allow a prover to demonstrate knowledge of a secret while revealing no information whatsoever
  - **beyond what the verifier was able to deduce prior to the protocol run**
- ZK protocols are often instances of interactive proof system, wherein a prover and verifier exchange multiple messages (challenges and responses), typically dependent on random numbers
- Example
  - **Fiat-Shamir identification protocol**

Basic Fiat-Shamir identification scheme

- It allows one party, Peggy (P), to prove to another party, Victor (V), that she possesses secret information without revealing the secret to V
  - **asymmetric cryptography identification scheme**
  - **it uses modular arithmetic**
- Setup
  - **a RSA-like modulus $n = pq$, is selected and published by the claimant P or by a trusted center T selects, while primes p and q are kept secret**
  - **each P selects a secret $s<n$ coprime to n, computes $v = s^2 \bmod n$, and publishes v (or v is sent to V)**
- Procedure
  - **each of t rounds has three messages as follows**
    - $P \rightarrow V$: $x = r^2 \bmod n$ (witness)
    - $P \leftarrow V$: $c \in \{0,1\}$ (challenge)
    - $P \rightarrow V$: $y = r\ s^c \bmod n$ (response)
- Verification (each round):
  - **V verifies that $y^2 = x\ v^c \bmod n$**

Fiat-Shamir identification scheme (cont.)

- Explanation:
  - **the challenge (or exam) *c* requires that P is capable of answering two questions, one of which demonstrates her knowledge of the secret *s*, and the other an easy question (for honest provers) to prevent cheating**
    - an adversary impersonating P might try to cheat by selecting any r and setting $x = r^2/v$, then answering the challenge $c = 1$ with a “correct” answer $y = r$; but would be unable to answer the exam $c = 0$ which requires knowing a square root of $x \bmod n$
    - a prover P knowing s can answer both questions, but otherwise can at best answer one of the two questions, and so has probability only 1/2 of escaping detection
  - **by iterating the protocol t times (e.g., $t = 20$ or $t = 40$), the probability of cheating decreases to an arbitrary acceptable small value $(1/2)^t$**
    - V accepts P’s identity only if all t questions (over t rounds) are successfully answered
- Security
  - **the security relies on the difficulty of extracting square roots modulo large composite integers *n* of unknown factorization, which is equivalent to that of factoring *n***

Authentication attacks

- Possible authentication attacks are:
  - **Impersonation attack**
    - pretend to be client or server
  - **Replay attack**
    - a valid message is copied and later resent
  - **Reflection attack**
    - re-send the authentication messages elsewhere
  - **Modify attack**
    - modify messages between client and server
  - **Compromising of key material**
    - steal client/server authentication database

Replay and reflection countermeasures

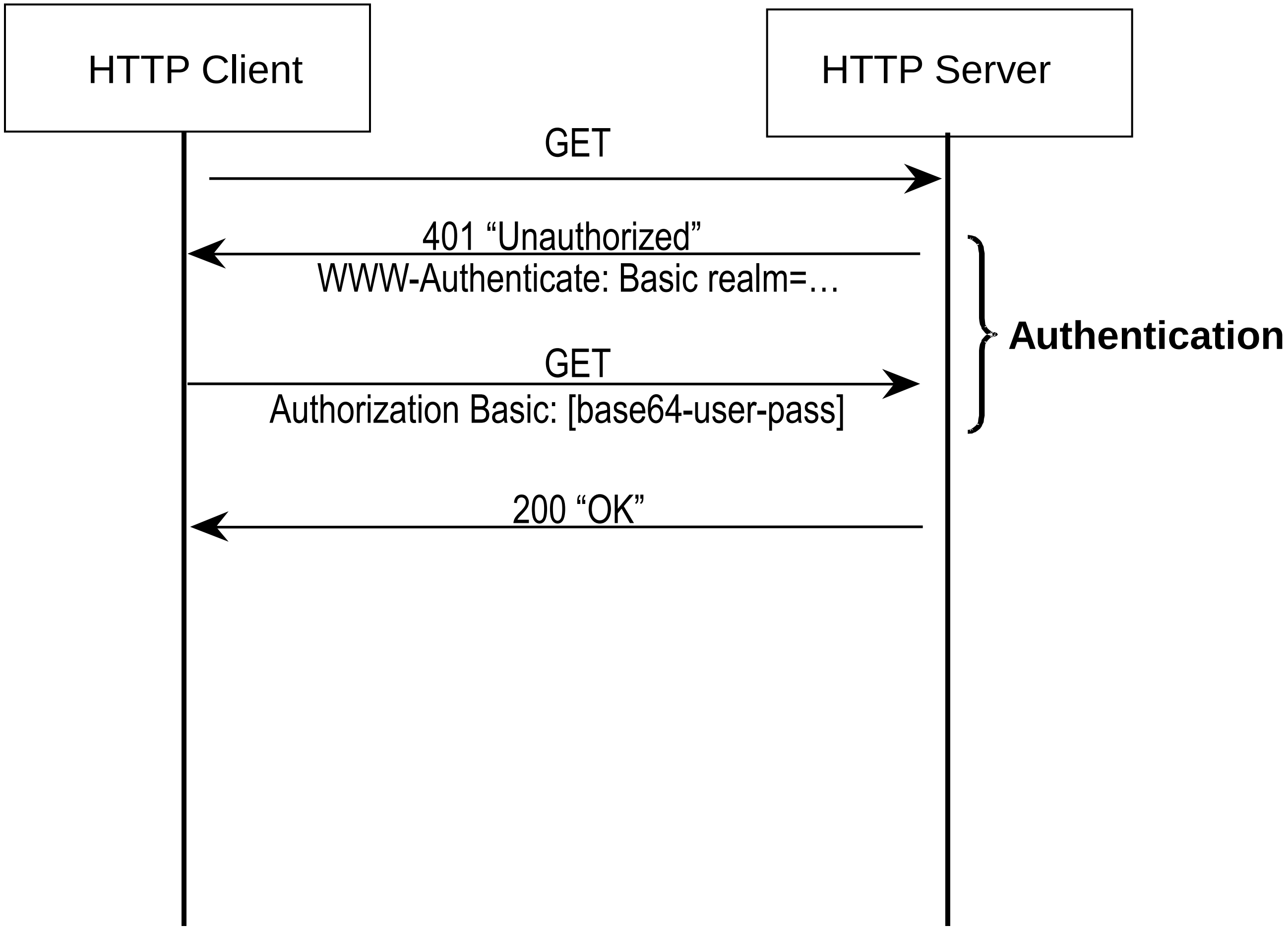
- Countermeasures against replay and reflection attacks include
  - **use of sequence numbers**
    - difficult to implement in practice
  - **use of timestamps**
    - needs synchronized clocks
  - **use of challenge/response**
    - using unique nonce, salt, realm values

Examples of authentication protocols

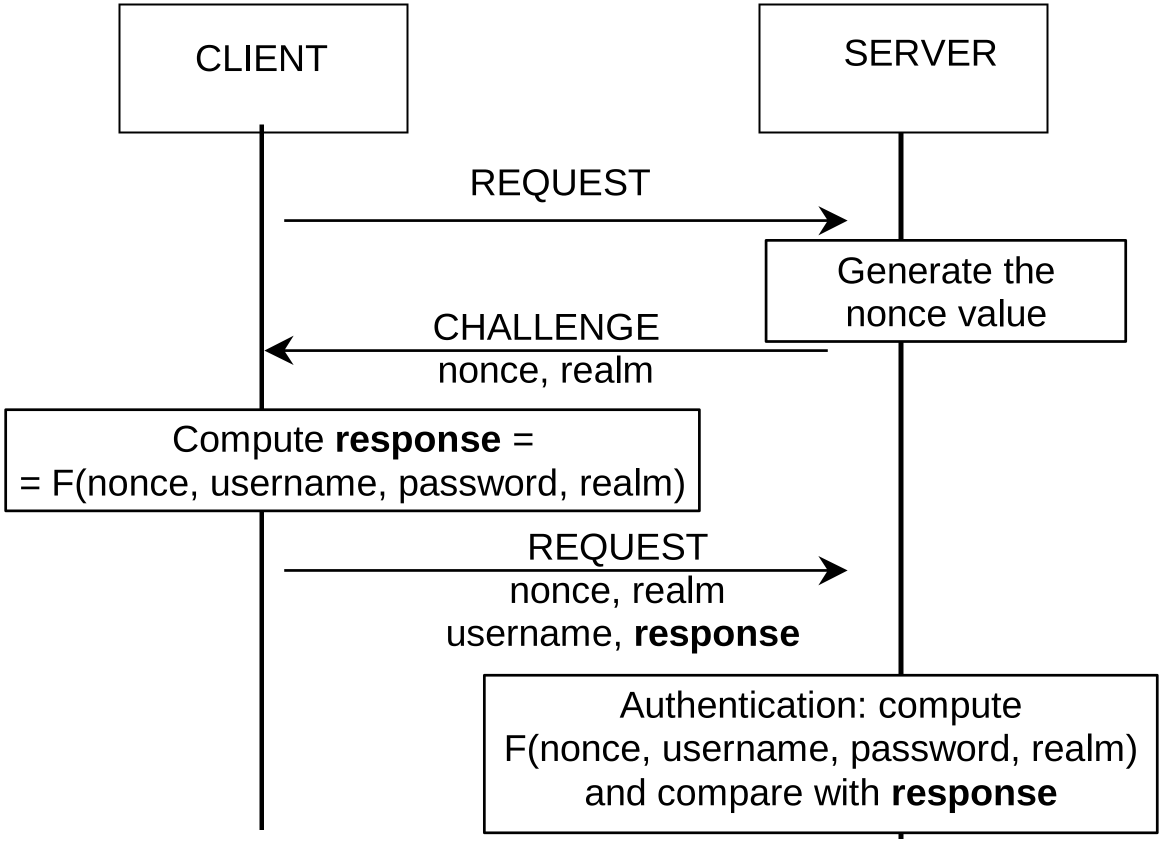
HTTP Basic and Digest Authentication

- Any time that a HTTP server receives a request, it MAY challenge the initiator of the request to provide assurance of its identity
- Two different types of client authentication schemes:
  - **“basic” authentication**
    - the client must authenticate itself with a user-ID and a password for a given realm
    - this scheme is not considered to be a secure method of user authentication as the user name and password are passed in an unencrypted form (unless secure transport is used)
  - **“digest” authentication**
    - based on a simple stateless challenge-response paradigm
    - the server challenges the client using a nonce value
    - a valid response contains a checksum (by default, through MD5) of the username, the password, the given nonce value, the HTTP method, and the requested URI
    - message authentication and replay protection
    - used also by other HTTP-based protocol (e.g. SIP for VoIP)

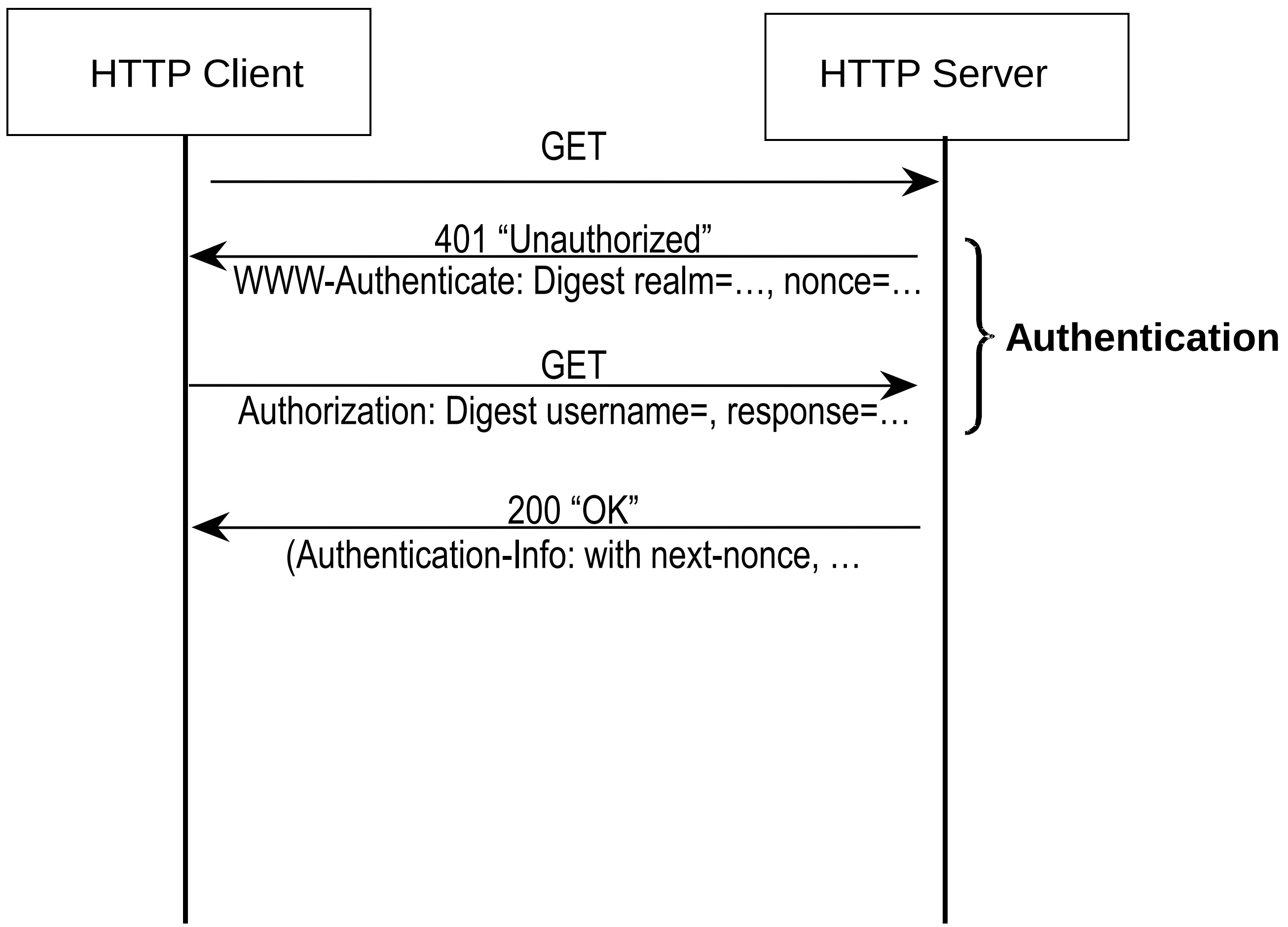
HTTP Basic authentication - Example



HTTP Digest authentication (cont.)



HTTP Digest authentication - Example



HTTP Digest authentication - Example (cont.)

- 401Unauthorized response message:

```
WWW-Authenticate: Digest realm="biloxi.com", qop="auth,auth-int",
          nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
          algorithm=MD5
```

- Next request message:

```
Authorization: Digest username="bob", realm="biloxi.com",
          nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
          uri="/dir/index.html", qop=auth,
          response="6629fae49393a05397450978507c4ef1"
```

➢ **response = F(nonce, username, passwd, realm, metod, http uri)**

Digest calculation details

- If the "qop" value is "auth" or "auth-int“, the F() digest value is computed as follows:
  - **If the "qop" directive's value is "auth" or is unspecified, then A2 is:**

    A2 = Method : digest-uri-value
  - **If the "qop" value is "auth-int", then A2 is:**

    A2 = Method : digest-uri-value : H(entity-body)
  - **If the "algorithm" directive's value is "MD5" or is unspecified, then A1 is:**

    A1 = username-value : realm-value : passwd
  - **then, F() is:**

    H(H(A1) : nonce-value : nc-value : cnonce-value : qop-value : H(A2))